Claims 1-7 were submitted for examination. The examiner objected to the specification as failing to provide an adequate written description of the claimed invention and all claims were rejected under 35 U.S.C. 112, first paragraph on that basis. All claims were rejected under 35 U.S.C. 112, second paragraph as being indefinite for failing to particularly point out and distinctly claim the invention. Claim 7 was rejected under 35 U.S.C. 112, fourth paragraph as being of improper dependent form. Claims 1-4 were rejected under 35 U.S.C. 102(b) as being anticipated by Deitel ("Operating Systems") and claims 5-7 were rejected under 35 U.S.C. 103 as being unpatentable over Deitel. All claims have been amended. Support for the amendments is found throughout the specification and figures.

<u>35 U.S.C. 112 Rejections</u>

35 U.S.C. 112, first paragraph

As noted in the Description of the Prior Art section of the specification, the design, control and use of both system calls and fast kernel traps are known and understood in the prior art. Applicant's invention is a novel method for using both to provide an overall improved system of operation. The specification and figures are adequate to disclose the invention to a person skilled in the art of operating system kernel programming.

35 U.S.C. 112, second paragraph

In response to the Examiner's remarks, claims 1-3, 5 and 6 have been amended.

Claim 1 was rejected on three grounds. As to rejection i, Applicant does not agree that the claim is vague for failure to identify which component causes the complication. Complications are discussed in the specification, for example at page 3, line 16 to page 4, line 14 and page 8, line 17 to page 9, line 14. A "complication" is any condition, regardless of source, that requires an action beyond the limited capabilities of the kernel function call. The complications mentioned are not new, just the method for dealing with them. A person skilled in the art would understand the specific limitations and capabilities of the kernel function calls and system calls in the particular operating system they are using and would understand the various sources of complications in that particular operating system.

As to rejections ii and iii of claim 1, the claim has been amended to indicate the processing path in the absence of a complication and to delete the phrase "handling the complication".

Claim 2 was rejected as confusing. In addition to the other amendments to claim 1 discussed above, claim 1 was also amended such that it applies to a method of operation during execution of a processing thread, not just during execution of a kernel function call. In view of amended claim 1, it is believed that claim 2 is now clear.

Claim 3 was rejected because "the step" lacked proper antecedent basis. Claim 3 has been amended to insert the word "additional" prior to "step" at both occurrences.

Claim 4 was rejected as confusing. Specifically, the examiner states the steps are redundant because "they must have been previously stored in the system for use by the kernel." The Examiner's requested amendment to claim 4 has not been adopted. Claim 4 does not require amendment because, as discussed in the specification at page 11, line 19 to page 12, line 3, the parameters passed to the system call includes at least one parameter, the KFC phase identifier, which is determined by the amount of processing the kernel function call has completed and, therefore, could not have been earlier stored in the system for use by the kernel.

As to the examiner's second objection re claim 4, the steps in claim 4 are the components of step (c)(1), not additional steps. Therefore, claim 4 is not redundant.

Claim 5 has been amended to delete the references to "phase" and "LWP".

Claim 6 has been amended to clarify that the steps listed are a more detailed description of step 1(c)(2), not additional steps.

35 U.S.C. 112, fourth paragraph

Since claim 1 now refers to processing thread execution rather than kernel function call execution, the step of releasing the spin lock in claim 7 is a proper limiting step.

35 U.S.C. 103 Rejections

Claim 1

Claim 1 was rejected under 35 U.S.C. 102(b) as being anticipated by Deitel.

What Deitel discloses is simply an interrupt handler. As discussed on pages 61-62, the Deitel interrupt handler is a special purpose piece of software designed to respond to external events. The Deitel interrupt handler determines the cause of the interrupt and passes the processing to a specific system process designed to handle that type of interrupt. While the Deitel interrupt handler is active, further interrupts are disabled.

By contrast, the kernel function call is a form of supervisor call that is invoked as a direct response to a user program request for a specified operation. The specified operation can be performed either by a kernel function call or by a system call. The kernel function call is desirable because it is smaller and faster than the system call. The drawback to the kernel function call is its lack of ability to handle complications.

Unlike Deitel's interrupt handler, interrupts are not disabled during kernel function call operations. In fact, claim 1 specifically claims monitoring for an interrupt during kernel function call execution and, if one is detected, promoting to a system call which can both deal with the interrupt and continue to do the program operation processing.

Claim 2

The examiner rejected claim 2 under 35 U.S.C. 102(b) on the basis that Deitel "teaches monitoring for a suspended state" and "teaches demoting from the system call to the kernel call".

Suspension and resumption of a process is a well known operating system principle. The typical prior art application of this principle is that a process will suspend and later will resume execution in the same state as it was prior to suspension. This is the context in which Deitel mentions suspension and resumption. There is no language in Deitel which in any way describes or suggests the idea of suspending process execution as a system call and later resuming that process execution as a kernel function call.

Claim 3

Claim 3 was rejected under 35 U.S.C. 102(b) on the basis that "Deitel teaches assigning a stack to the system call and releasing the stack as creating and destroying the stack of a process created by a system call."

Deitel does not discuss stacks associated with system calls. Paragraph 5 on page 575 of Deitel simply mentions that each process has a "stack region" which contains the process's user stack. As Deitel indicates, a process may later initiate a system call (paragraph 6, page 575). Deitel teaches nothing about how kernel stacks may or may not be assigned or released in connection with system calls and nothing in Deitel in any way describes or suggests the method claimed in claim 3. Deitel has no notion of the idea of promotion/demotion between a kernel function call and a system call and certainly no notion of the specific steps of claim 3: assigning a kernel stack at the time of promotion to a system call and releasing the kernel stack at the time of demotion to a kernel function call.

Claim 4

Claim 4 was rejected under 35 U.S.C. 102(b) as being anticipated by Deitel. Specifically, "The rejection above in section i applies since the interrupted parameters must be stored in the computer in order for the system call to be initiated."

The exact basis of the examiner's rejection is not clearly understood. If the examiner is saying that certain parameters must be passed to the kernel at the initiation of the kernel function call and, therefore, these parameters are already available to the system call, the following comments are offered. As discussed on page 11, line 19 to page 12, line 2, the parameters stored in claim 4 are <u>not</u> identical to the parameters that were passed from user space to the kernel function call. The parameters to be passed to the system call include the KFC phase identifier. As discussed at page 12, lines 7-18, the system call does not need to repeat processing already performed by the kernel function call, therefore the system call must receive an indication of how far along the kernel function call was at the time of promotion. Obviously, this is new information that was not available at the time of kernel function call initiation.

<u>Claims 5 and 6</u>

Claims 5 and 6 were rejected under 35 U.S.C. 103 as being unpatentable under Deitel.

The examiner is again equating the operation of the Deitel interrupt handler with the method claimed by Applicant. An interrupt handler as described by Deitel is designed to do the minimum amount of processing possible to identify an interrupt and then pass control to another process. The interrupt handler cannot not pass control <u>during</u> its processing, therefore Deitel has no notion of passing or checking an "identifier" as claimed in amended claims 5 and 6.

Also, claims 5 and 6 are not directed at the typical suspension and resumption. Applicant's claimed method is directed at passing control of the execution of a process between alternative means of performing the execution depending on whether or not capabilities beyond those needed for the basic execution of the process are required.

Neither the kernel function call nor the system call is analogous to the Deitel interrupt handler. Each kernel function call and its corresponding system call are designed to perform an operation in response to a user program. Performing the operation with the kernel function call is the option of first choice, since the kernel function call is smaller and faster than the system call. If no complications are detected, the kernel function call will complete the operation requested and return. If a complication is detected, the system call is activated. The system call will deal with the complication and will also pick up the execution of the desired operation at the proper point indicated by the identifier passed by the kernel function call. Execution of the system call continues until either the operation is complete or a suspend state is reached. If a suspend state if reached by the system call prior to completion of the operation, execution of the desired operation is transferred back to the kernel function call for completion. The kernel function call picks up execution of the operation at the appropriate point as indicated by the identifier passed back by the system call. Deitel does not in any way show or suggest this method.

Claim 7

This claims depends from amended claim 1. In the context of the novel method of claim 1, the step of releasing the spin lock is not obvious.

The Examiner also cited Laggis et al (U.S. Patent 5,109,515), Flurry (U.S. Patent 5,201,608), Tajalli et al (U.S. Patent 5,361,359), Orton et al (U.S. Patent 5,379,432), Gaertner et al (U.S. Patent 5,390,329) and Seong Rak Rim et al "Message-based Microkernel for Real-Time System" as being pertinent to the disclosure. None of these references is read as showing or suggesting the claimed methods.

In summary, none of the cited references viewed either alone or in combination disclose or suggest the particular claimed methods in amended claims 1-7. Favorable action on all claims is respectfully requested.

If any additional fee is required by the filing of this amendment, the Commissioner is hereby authorized to charge Deposit Account 04-0165.

Date: September 7, 1995

Respectfully submitted,

[illegible]

Robert L. Dulaney
Reg. No. 28,071
Data General Corporation
4400 Computer Drive
Westboro, MA 01580
(508) 898-6220

<u>Certificate of Mailing</u>

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner of Patents and Trademarks, Washington, D.C. 20231, on September 7, 1995.

Registered Representative: Robert L. Dulaney

Signature: [illegible]

Date of Signature: Sept. 7, 1995